Computer game engines for developing first-person virtual environments

David Trenholme and Shamus P. Smith*
Durham University
Durham DH1 3LE
United Kingdom

March 14, 2008

1 Introduction

Building realistic virtual environments is a complex, expensive and time consuming process [Lai02, RBFR03, SD00, SH01]. Lepuras and Vassilakis [LV05] note that state-of-the-art virtual environments are both costly to build and to maintain. In addition to generating virtual object models [Kes02, SW06], for example buildings and scenery, a developer must also manage support for user interaction [BKLP05, SH06], any environment and object behaviours that are required, for example collision detection, and any non-visual features of the virtual world, for example audio cues and haptic feedback [PBT02]. Although virtual environment development toolkits are available, many only provide a subset of the tools needed to build complete virtual worlds. Some features of virtual worlds such as wind, fire, smoke and water, and the provision for embodied autonomous agents [AB02], are particularly hard to simulate. In addition, virtual environment toolkits often require additional programming skills and a substantial time investment on the part of the developer.

The current generation of computer games present realistic virtual worlds featuring user friendly interaction and the simulation of real world phenomena, for example gravity. Using computer games as the basis for virtual environment development has a number of advantages. Computer games are robust and extensively tested [LV05], both for usability and performance, work on off-the-shelf systems [RBFR03] and can be easily disseminated, for example via online communities. Many computer game developers support modification of their game environments by releasing level editors, for example to modify the game environment, and tools to edit the game behaviour. This allows the reuse of the underlining game engine technology, including 3D rendering, 2D drawing, sound, user input and world physics/dynamics [LJ02]. Therefore computer games can provide inexpensive state-of-the-art 3D virtual worlds that can be customised to experimental requirements in a short amount of time without extensive programming. This is a promising area of research that has already seen recent game technology used in a variety of virtual environment domains including context-aware system evaluation [BE01, OLMD07], e-Tourism[1] [BMS07], human behaviour model testing [SBOC06], human-level AI research [Lai02, LAB+02], human-robot interaction simulation [LWH07, WLG03], information visualisation [KWGH05], interactive storytelling [CCM02], laboratory accident simulation [BF03], landscape visualisation [HP02], large-scale real-time ecosystem simulation [ROB02], phobia therapy [BCSJ+06, RBFR03], photorealistic environment walk-throughs [DB00], psychological experimenting [FHK+07], serious games [MRL+06] and virtual museums [LV05].

A number of computer game developers provide tools, documentation and source code, either with the game itself or separately available, so that end-users can create new content for the game called a *mod* or *modification* [Gui07]. For example, users can create new levels, maps, items or characters and

*Contact author - shamus.smith@durham.ac.uk, tel. +44(0)191 334 4284, fax +44(0)191 334 1701

[1]The e-Tourism environment developed by Berger et al. [BMS07] uses the Torque Game Engine (see http://www.garagegames.com/ [last access 17/7/07]). However, this game engine requires a commercial licence for non-game development and will not be discussed here further.

add them into the game, known as a *partial conversion*, or create an entirely new game by altering the games source code, known as a *total conversion*.

This report overviews several currently available game engines that are suitable for prototyping virtual environments. Modern game engines have a modular structure so that they can be reused for different games [LJ02]. Therefore, a game produced using an engine can be modified using the development tools provided with it in order to produce a virtual environment. Computer games in the First Person Shooter (FPS) genre tend to have the greatest capability and resources for modification, so this report will examine game engines for this game genre. There is a wide selection of 3D game engines available for potential reuse[2]. Lewis and Jacobson [LJ02] observe that there are more than 600 commercial game engines. However, the game engines considered here are those used in the most recently available commercial computer games, thus representing more advanced technology which usually results in more realistic environments. The Quake III Arena engine (see Section 3), although not as recent as the others, is also considered as it is different to the engine used for Quake IV, now the Doom 3 engine (see Section 4) and is still popular for modding.

2 CryENGINE

The *CryENGINE* was created by software developers Crytek[3] and used in the game Far Cry[4] released in 2004.

The CryENGINE supports a number of features that are useful for creating immersive and realistic games and virtual environments, such as a real-time editor, bump mapping, dynamic lights, a network system, an integrated physics system, shaders, shadow support and a dynamic music system. The necessary development tools are integrated with the engine itself, including the *CryENGINE Sandbox* world editing system. The engine supports all currently available hardware and it is updated with further hardware support when it becomes available. Licensed developers receive full source code and documentation for the engine and tools[5].

Advantages to using the CryENGINE include the fact that the engine produces very high quality graphics and visuals. The necessary development tools are provided with the engine and so can be accessed from games that use the engine, such as Far Cry. Also, the Sandbox editor is a very intuitive tool as it edits levels in real time, offering *what you see is what you play* feedback. Partial source code and documentation is included with a freely downloadable SDK[6]. Disadvantages with this engine include that it has high demands on supporting hardware requirements for the high quality visual and audio components and that there is only a small amount of fan-based community written documentation available. This has implications for getting support if this engine is used in virtual environment development. However, there is a growing mod community at the official *Crymod Modding Portal*[7].

3 id Tech 3 Engine

The *id Tech 3 engine* is a game engine developed by id Software[8] and was first used in the game Quake III Arena[9], released in 1999. It was previously known as the Quake III engine. The engine was the successor to the Quake[10] and Quake II[11] engines, but much of the code is either new or has been re-written. Designed to be the ultimate multiplayer experience, QUAKE III Arena has become a defacto standard for professional gamers and is the common choice for gaming tournaments around the world[12]. This game engine has been succeeded by the Doom 3 engine (see Section 4).

[2]For a comprehensive database of 3D engines see http://www.devmaster.net/engines/ [last access 17/7/07].
[3]http://www.crytek.com [last access 14/7/07].
[4]http://www.farcry-thegame.com [last access 14/7/07].
[5]http://www.crytek.com/technology/index.php [last access 14/7/07].
[6]http://crymod.com/filebase.php [last access 18/7/07].
[7]http://crymod.com/ [last access 18/7/07].
[8]http://www.idsoftware.com [last access 14/7/07].
[9]http://www.idsoftware.com/games/quake/quake3-arena/ [last access 14/7/07].
[10]An augmented reality gaming system based on the Quake engine is described in [PT02].
[11]Network simulations using code based on the Quake II engine are described in [SA06]. Research into human-level AI using Quake II is described in [Lai02].
[12]http://www.idsoftware.com/business/history/ [last access 14/7/07].

The id Tech 3 engine supports 3D models in the MD3 file format, which uses per-vertex animation to store movement information, instead of skeletal animation. The MD3 format defines models in three separate parts, e.g. head, torso and legs, so that each part can move independently. Each part of the model has its own texture set. Character models are lit and shaded using gouraud shading[13] while the map levels use lightmaps or gouraud shading, depending on the user's preference. Three different kinds of shadow are supported: *blob shadow*, a circle with faded edges at the feet of a character, and two other modes, which project an accurate shadow on the floor - the difference being that one mode is opaque while the other mode attempts to project shadow volumes in a medium-transparent black. However, none of these techniques clip shadow volumes, resulting in shadows passing through geometry. A high-level shader language is also included, as well as a method for rendering effects such as volumetric fog[14].

One of the advantages of using this engine is that in addition to the Quake III game source code, the engine source code has been released under the GNU General Public License (GPL). The availability of the full source code provides more flexibility in the customisation of game environments. Additionally there is a large number of tutorials and articles that have been written for this engine and with the release of the source code[15] in 2005, there is still an active development community. One disadvantage of using this engine is that game engine technology has progressed greatly since its original release in 1999, and more advanced engines are available for use [KWGH05]. The more advanced engines can help with realism and immersion in a virtual environment because of their greater graphical capabilities, for example, in the generation of realistic shadows and lighting effects, and in game customisation with more developer-oriented support tools.

Virtual environments developed using this engine include environments for context-aware system evaluation [BE01], information visualisation [KWGH05] and psychological experimenting [FHK+07].

4 id Tech 4 Engine

The *id Tech 4 engine* is a game engine developed by id Software and was first used in game Doom 3[16]. id Tech 4 was previously known as the Doom 3 engine. The Doom 3 engine began as an enhancement of the Quake III engine, which consisted of a complete rewrite of the renderer. However, it was decided to change from the C programming language to C++, which required a complete restructuring and rewrite of the rest of the engine[17].

The id Tech 4 engine provides several unique features, such as a unified lighting paradigm, where every surface uses the same rendering pipeline, leading to visual consistency and fully dynamic shadows. An in-game GUI feature is also provided, which enables interactive, resolution-independent surfaces, such as the computer screens in Doom 3. The engine also makes short map compilation times possible. The level editor for this engine is built into games that use it and so can be accessed by anyone who owns any of these games. Greater flexibility is enabled when modding by the use of non-proprietary formats. Finally, the engine uses a strong and flexible scripting language, which can accomplish effects from changing entity behaviour to modifying variables in real time[18].

The main advantage of using the Doom 3 engine is that it is a very powerful engine, capable of producing realistic environments. Also, there is a large modding community, including a large number of tutorials and articles are available. However, similar to CryENGINE (Section 2), the hardware requirements of the engine are high, for example requirements for Doom 3 include: 100% DirectX 9.0b compatible 64MB hardware accelerated video card, Pentium IV 1.5 GHz or Athlon XP 1500+ processor or higher, 384MB RAM, 2.2GB of uncompressed free hard disk space (plus 400MB for Windows swap file) and 100% DirectX 9.0b compatible 16-bit sound card.

[13] Gouraud shading is a method used in computer graphics to simulate the differing effects of light and colour across the surface of an object to eliminate intensity discontinuities [FvFH90, pg736].

[14] http://en.wikipedia.org/wiki/Quake_III_engine [last access 14/7/07].

[15] The Quake III source code and development tools are available at http://www.idsoftware.com/business/techdownloads/ [last access 14/7/07].

[16] http://www.idsoftware.com/games/doom/doom3/ [last access 14/7/07].

[17] http://en.wikipedia.org/wiki/Doom_3_Engine [last access [14/7/07]

[18] http://www.modwiki.net/wiki/Doom_3_engine [14/7/07].

5 Jupiter Extended (EX)

Jupiter EX is the latest version of a multi-platform game engine and development kit, developed by Touchdown Entertainment[19] and used for the game F.E.A.R[20].

Visual technologies used in the engine include a DirectX 9 based renderer that uses materials for all objects. Shaders are associated with each material and provide editable parameters, including texture maps, colours and numeric constants. For lighting, Jupiter EX uses a unified Blinn-Phong per-pixel lighting model[21], which allows each light to generate both diffuse and specular lighting consistently across all solid objects in the environment. The lighting pipeline uses three passes: emissive, lighting and translucency. The engine also supports a data driven visual effects system, which allows the creation of key-framed effects that can be comprised of dynamic lights, particle systems, models and sounds. For example, particle effects such as fire, steam and smoke can be produced. Havok physics[22] have been incorporated into this version of the engine to simulate realistic physics and the Havok Vehicle Kit is included to support vehicles and simulate vehicle behaviour. A flexible animation system allows control of character movement and facial expressions[23].

Jupiter EX includes a set of content creation tools, each with a different purpose. *WorldEdit* allows designers to create 3D environments and add objects and lighting. The results can be viewed in real time. *ModelEdit* allows the designer to optimise models so that they interact correctly with the environment and other objects. It is also possible to define specific properties, assign texture IDs, attach objects to models and manipulate animation data. *FxEdit* can create and modify visual effects, without requiring re-compilation of the game, which increases the efficiency of the implement/test cycle for special effects [24]. Other tools include *ArchiveEdit*, a tool used to package game assets into one archive for distribution, *GDBEdit*, the game database editor for changing things such as weapon damage, the interface etc. without changing source code, *StringEdit*, used to store and modify all strings that need to be localised and a number of plug-ins for importing/exporting to modelling applications[25].

The advantages of modifying games that use the Jupiter EX engine include the high level of graphical fidelity that can be achieved, which will improve realism and immersion. Also, the tools required for modifying Jupiter EX games are logically separated into different applications and come with documentation. The main disadvantage is that the modding community is not as large for F.E.A.R, the most popular Jupiter EX game, as it is for other games such as Half-Life 2 (see Section 6) or Unreal Tournament 2003/2004 (see Section 7). Therefore there are fewer community websites and fan written tutorials available. Although documentation is included with the F.E.A.R SDK, it is not comprehensive, and this has time implications for learning how to create environments and achieve certain modifications.

6 Source Engine

The *Source Engine* was developed by the Valve Corporation[26] and is most notably used in games such as Half-Life 2[27] and Counter-Strike: Source[28]. The Source Engine features include a high degree of modularity and flexibility, lip synchronisation and facial expression technology and a realistic physics system.

The engine also supports particle effects, volumetric smoke and environmental effects such as fog or rain. An advanced artificial intelligence system is provided which enables sophisticated character navigation, enabling characters to run, jump, climb stairs etc. The source code is written in C/C++ and classes can be modified or derived to create new entities, allowing a finer degree of control over modifications. A high level of graphical fidelity can be achieved using the Source Engine because of its advanced rendering features and detailed characters. Also, the materials system can be used to define

[19] http://www.touchdownentertainment.com/ [last access 14/7/07].
[20] http://www.whatisfear.com [last access 14/7/07].
[21] For details on *Blinn* and *Phong* lighting models see [Ebe07, pg96-97].
[22] http://www.havok.com/ [last access 18/7/07].
[23] http://www.touchdownentertainment.com/jupiterEX.htm [last access 14/7/07].
[24] http://www.touchdownentertainment.com/jupiterEX.htm [last access 14/7/07].
[25] ftp://ftp.sierra.com/pub/sierra/fear/updates/ [last access 14/7/07].
[26] http://www.valvesoftware.com [last access 15/7/07].
[27] http://www.Half-Life2.com/ [last access 15/7/07].
[28] http://counter-strike.net/ [last access 15/7/07]

not only what texture an item uses, but also what it is made from, which will affect the sound it makes when moved and its mass. All these features will improve the realism of a virtual environment[29].

The level editor that is used to produce levels for games using the Source Engine is the *Valve Hammer Editor*, commonly referred to as *Hammer*. The editor uses brushes, called primitives, to construct a level. Apart from the construction of level architecture, Hammer is also used for creating level events and scripting[30]. Hammer is an additive-based level editor and uses CSG (constructive solid geometry) operations to add geometry to a void.

Documentation for the Source SDK is provided on the Valve Developer Community[31] wiki, which was created by Valve and is the definitive and most comprehensive source of information on using the Source Engine. All the necessary tools for creating a mod and game content, e.g. via Hammer, are included with the Source SDK, which is available after purchasing a Source-based game, i.e. Half-Life 2, and can be downloaded using Valve's digital content delivery service called *Steam*[32]. The Source SDK also provides a *Create a Mod* option, which copies the necessary source code and resources to a working directory. A large number of other websites provide their own discussion forums and tutorials, which range from general introductions to the Source SDK tools to more specific tasks, see for example *Interlopers.net*[33], for design resources and Half-Life 2 tutorials, and *Edit Life*[34], for Half-Life 2 Source mapping tutorials.

The main advantage of the Source Engine is that it has been designed from the ground up to be highly modular, which means that modification of games developed using the Source Engine can be done to a very fine degree. Also, the popularity of the Source Engine for game modding has led to a very large community of licensees and independent developers, who participate in forum discussions, write articles and create tutorials for the rest of the community to learn from. The Valve Developer Community provides tutorials and technical references relevant to developing mods for Source-based games, such as Half-Life 2. One of the disadvantages of the using the Source Engine is that the user will have to have a Steam account to be able to use the mod. However, Steam accounts are free but initially require an internet connection.

Virtual environments developed using the Half-Life and Source Engine include systems to evaluate adaptive context-aware services [OLMD07], arachnophobia therapy [BCSJ+06, RBFR03], laboratory accident simulation [BF03] and serious games to teach food safety [MRL+06].

7 Unreal Engine 2

The *Unreal Engine* is a powerful and widely used game engine developed by Epic Games[35], the second version of which was used to develop Unreal Tournament 2003[36] and its successor, Unreal Tournament 2004[37]. The core code is written in C++ and has been used in games on several platforms. *UnrealEd 3*, the level editor, is included with Unreal Tournament 2003/2004.

The engine supports high performance rendering, advanced animation features and high-quality dynamic lighting. Supported effects include a particle system for particles composed of sprites, meshes, lines and beams. The particle system supports various lifetime, texture, movement and collision options. All particle features can be manipulated in real time in the editor. The texturing features of the engine include a material system that supports alpha-blending, i.e. transparency and blending of multiple layers of textures. High resolution textures that use efficient compression are also supported, which can help improve realism. The Unreal Editor, *UnrealEd*, is a *what you see is what you get* content creation tool[38].

In contrast with other game engine level editors, UnrealEd regards the game virtual world as a solid mass. UnrealEd is therefore a subtractive-based level editor where virtual worlds are carved from the solid mass. This has the advantage of removing any possibility of BSP[39] leaks, caused by gaps between geometric primitives, when constructing environment spaces, as can happen when using Valve's Hammer,

[29] http://www.valvesoftware.com/files/SOURCE_InfoSheet.pdf [last access 15/7/07].
[30] http://developer.valvesoftware.com/wiki/Hammer [last access 15/7/07].
[31] http://developer.valvesoftware.com/ [last access 15/7/07].
[32] http://www.steampowered.com [last access 15/7/07].
[33] http://www.interlopers.net/ [last access 15/7/07].
[34] http://www.editlife.net/ [last access 15/7/07].
[35] http://www.epicgames.com/ [last access 16/7/07].
[36] http://www.unrealtournament3.com/ [last access 16/7/07].
[37] http://www.unrealtournament.com/ last access [16/7/07].
[38] http://www.unrealtechnology.com/html/technology/ue2.shtml [last access 16/7/07].
[39] *Binary space partitioning* [Ebe07, pg354].

as the UnrealEd virtual world is totally encapsulated. Modification of games based on the Unreal Engine 2 is further supported by inclusion of sample content and the source code for the engine and editor. The engine source code is written in C++. The engine also includes a scripting language, *UnrealScript*, which can modify various aspects of the gameplay[40].

Many resources are available for modifying Unreal Engine based games, including Epic's *Unreal Developer Network (UDN)*[41], which is the official support site for licensees and mod developers and provides technical documentation as well as tutorials for the Unreal Engine and UnrealEd. There are also a number of community websites that provide discussion forums and tutorials, for example *Architectonic UT2003 Editing and Inspiration*[42] and *UnrealWiki*[43].

The main advantage of the Unreal Engine 2 is that it has been available for a few years so there are a considerable number of tutorials and articles on editing games that have been developed using it. The main disadvantage is that there are now more recent engines with more advanced graphics technology that will help to improve the realism of developed virtual environments.

Virtual environments developed with the Unreal Engine include systems for acrophobia and claustrophobia therapy [RBFR03], virtual museums [LV05], interactive storytelling [CCM02], landscape visualisation [HP02], large-scale real-time ecosystem simulation [ROB02], human behaviour model testing [SBOC06], human-level AI research [Lai02, LAB+02], human-robot interaction simulation [LWH07, WLG03], photorealistic walk-throughs of the Florida Everglades and the Notre Dame Cathedral [DB00] and a low cost CAVE [Jac03].

8 Summary

Much of the research and development being conducted in the gaming industry parallels ongoing work in the virtual environment community [Zyd05]. Gershon et al [GSP06] observe that computer games fuse software engineering, architecture, artificial intelligence, 3D graphics, art and sound effects with dramatic performances, music and storytelling. Many of these features are required in virtual environments and reusable game engines have the potential to support this transition of technology.

All the engines considered in this report are suitable for developing non-game virtual environments. Lewis and Jacobson [LJ02] note that in many cases it would be hard to imagine an application for which one game engine would be suited and others not. All the engines listed provide some source code to users. This is usually partial source code that can be altered and recompiled to affect the gameplay. Full source code, e.g. code for the engine, is typically only available to commercial licensees. However, this does change over time as new game engines are developed and older engines are released as open source projects. The existence of a strong and active online modding community, particularly with official developer support, is an important consideration, as is the minimum required specification for off-the-shelf hardware. Appendix A compares and contrasts some of the important features that relate to modification and editing of games using the engines considered in this report.

9 Acknowledgements

This work was funded in part by the Nuffield Foundation (Grant URB/34118)[44].

References

[AB02] Jan M. Allbeck and Norman I. Badler. Embodied autonomous agents. In K. M. Stanney, editor, *Handbook of Virtual Environments*, pages 313–332. Lawrence Erlbaum Associates, 2002.

[40] http://www.unrealtechnology.com/html/technology/ue2.shtml [last access [16/7/07].
[41] http://udn.epicgames.com [last access 16/7/07].
[42] http://architectonic.planetunreal.gamespy.com/ [last access 16/7/07].
[43] http://wiki.beyondunreal.com/ [last access 16/7/07].
[44] http://www.nuffieldfoundation.org/ [last access 15/7/07].

[BCSJ+06] Stéphanie Bouchard, Sophie Côté, Julie St-Jacques, Geneviève Robillard, and Patrice Renaud. Effectiveness of virtual reality exposure in the treatment of arachnophobia using 3D games. *Technology and Health Care*, 14:19–27, 2006.

[BE01] Markus Bylund and Fredrik Espinoza. Using Quake III Arena to simulate sensors and actuators when evaluating and testing mobile services. In *Extended abstracts of the ACM conference on human factors in computing systems (CHI 2001)*, pages 241–242. ACM, 2001.

[BF03] John T. Bell and H. Scott Fogler. Implementing virtual reality laboratory accidents using the Half-Life game engine, WorldUp, and Java3D. In *2003 American Society for Engineering Education Annual Conference & Exposition*. American Society for Engineering Education, 2003.

[BKLP05] Doug A. Bowman, Ernst Kruijff, Joseph J. LaViola Jr., and Ivan Poupyrev. *3D User interfaces: Theory and Practise*. Addison Wesley, USA, 2005.

[BMS07] Helmut Berger, Dieter Merkl, and Simeon Simoff. Open new dimensions for e-Tourism. *Virtual Reality*, 11:75–87, 2007.

[CCM02] Marc Cavazza, Fred Charles, and Steven J. Mead. Emergent situations in interactive storytelling. In *Proceedings of the 2002 ACM Symposium on Applied Computing*, pages 1080–1085. ACM, 2002.

[DB00] Victor DeLeon and Robert Berry, Jr. Bringing VR to the desktop: Are you game? *IEEE MultiMedia*, April-June:68–72, 2000.

[Ebe07] David H. Eberly. *3D Game Engine Design: A practical approach to real-time computer graphics*. Morgan Kaufmann Publishers, Amsterdam, second edition, 2007.

[FHK+07] Andreas Frey, Johannes Hartig, André Ketzel, Axel Zinkernagel, and Helfried Moosbrugger. The use of virtual environments based on a medication of the computer game Quake III Arena in psychological experimenting. *Computers in Human Behaviour*, 23:2026–2039, 2007.

[FvFH90] James D. Foley, Andres van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, Reading, second edition, 1990.

[GSP06] Nahum Gershon, Ben Sawyer, and J. R. Parker. Games and technology: Developing synergy. *IEEE Computer*, pages 129–130, December 2006.

[Gui07] Erik Guilfoyle. *Half-Life 2 Mods for Dummies*. Wiley Publishing, Inc, 2007.

[HP02] Adrian Herwig and Philip Paar. Game engines: Tools for landscape visualization and planning? In *Trends in GIS and Virtualization in Environmental Planning and Design*, pages 161–172. Wichmann Verlag, 2002.

[Jac03] Jeffrey Jacobson. Using "CaveUT" to build immersive displays with the Unreal Tournament engine and a PC cluster. In *ACM Symposium on Interactive 3D Graphics*, pages 221–222. ACM Press, 2003.

[Kes02] G. Drew Kessler. Virtual environment models. In K. M. Stanney, editor, *Handbook of Virtual Environments*, pages 255–276. Lawrence Erlbaum Associates, 2002.

[KWGH05] Blazej Kot, Burkhard Wuensche, John Grundy, and John Hosking. Information visualisation utilising 3D computer game engines case study: A source code comprehension tool. In *6th ACM SIGCHI New Zealand chapter international conference on computer-human interaction (CHINZ 2005)*, pages 53–60. ACM, 2005.

[LAB+02] John E. Laird, Mazin Assanie, Benjamin Bachelor, Nathan Benninghoff, Syed Enam, Bradley Jones, Alex Kerfoot, Colin Lauver, Brian Magerko, Jeff Sheiman, Devvan Stokes, and Scott Wallace. A test bed for developing intelligent synthetic characters. In *Artificial Intelligence and Interactive Entertainment: Papers from the 2002 AAAI Spring Symposium*, pages 52–56. AAAI, 2002.

[Lai02] John E. Laird. Research in human-level AI using computer games. *Communications of the ACM*, 45(1):32–35, 2002.

[LJ02] Michael Lewis and Jeffrey Jacobson. Game engines in scientific research. *Communications of the ACM*, 45(1):27–31, January 2002.

[LV05] George Lepouras and Costas Vassilakis. Virtual museums for all: employing game technology for edutainment. *Virtual Reality*, 8:96–106, 2005.

[LWH07] Michael Lewis, Jijun Wang, and Stephen Hughes. USARSim: Simulation for the study of human-robot interaction. *Journal of Cognitive Engineering and Decision Making*, 1(1):98–120, 2007.

[MRL+06] B. Mac Namee, P. Rooney, P. Lindstrom, A. Ritchie, F. Boylan, and G. Burke. Serious Gordon: Using serious games to teach food safety in the kitchen. In *9th International Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games CGAMES06*, 2006.

[OLMD07] Eleanor O'Neill, David Lewis, Kris McGlinn, and Simon Dobson. Rapid user-centred evaluation for context-aware systems. In Gavin Doherty and Ann Blandford, editors, *Interactive Systems. Design, Specification and Verification. 13th Interactional Workshop, DSVIS 2006*, volume LNCS 4323, pages 220–233. Springer, 2007.

[PBT02] George V. Popescu, Grigore C. Burdea, and Helmuth Trefftz. Multimodal interaction modelling. In K. M. Stanney, editor, *Handbook of Virtual Environments*, pages 435–454. Lawrence Erlbaum Associates, 2002.

[PT02] Wayne Piekarski and Bruce Thomas. ARQuake: The outdoor augmented reality gaming system. *Communications of the ACM*, 45(1):36–38, 2002.

[RBFR03] Geneviève Robillard, Stéphane Bouchard, Thomas Fournier, and Patrice Renaud. Anxiety and presence using VR immersion: A comparative study of the reactions of phobic and non-phobic participants in therapeutic virtual environments derived from computer games. *CyberPsychology & Behaviour*, 6(5):467–475, 2003.

[ROB02] Scot Thrane Refsland, Takeo Ojika, and Robert Berry, Jr. Enhanced environments: Large-scale, real-time ecosystems. *Presence*, 11(3):221–246, June 2002.

[SA06] Anthony Steed and Cameron Angus. Enabling scalability by partitioning virtual environments using frontier sets. *Presence*, 15(1):77–92, February 2006.

[SBOC06] Barry G. Silverman, Gnana Bharathy, Kevin O'Brien, and Jason Cornwell. Human behaviour models for agents in simulators and games: Part II: Gamebot engineering with PMFserv. *Presence*, 15(2):163–185, April 2006.

[SD00] Shamus P. Smith and David J. Duke. Binding virtual environments to toolkit capabilities. *Computer Graphics Forum*, 19(3):C–81–C–89, 2000.

[SH01] Shamus P. Smith and Michael D. Harrison. Editorial: User centred design and implementation of virtual environments. *International Journal of Human-Computer Studies*, 55(2):109–114, 2001.

[SH06] Shamus P. Smith and Jonathan Hart. Evaluating distributed cognitive resources for wayfinding in a desktop virtual environment. In Y. Kitamura, D. Bowman, B. Fröhlich, and W. Stürzlinger, editors, *IEEE Symposium on 3D User Interfaces 2006*, pages 3–10. IEEE, 2006.

[SW06] Shamus P. Smith and James S. Willans. Virtual object specification for usable virtual environments. In *Annual Conference of the Australian Computer-Human Interaction Special Interest Group (OzCHI 2006)*. ACM, 2006.

[WLG03] J. Wang, M. Lewis, and J. Gennari. USAR : A game based simulation for teleoperation. In *47th Annual Meeting of the Human Factors and Ergonomics Society*, Denver, CO, 2003.

[Zyd05] Michael Zyda. From visual simulation to virtual reality to games. *IEEE Computer*, pages 25–32, September 2005.

A Engine Comparison

Table 1 compares and contrasts some of the important features that relate to modification and editing of games using the engines considered in this report.

Engine name	**Level editor**	**Source code**	**SDK**	**Documentation**	**Other**
CryENGINE	Sandbox, integrated with FarCry.	Game source code available, comes with CryENGINE MOD SDK.	CryENGINE MOD SDK, includes documentation and additional tools.	SDK includes CryENGINE modding guide and FAQ, as well as guides for other tools.	Uses a real-time sandbox editor. Paid license required for full source code and documentation.
id Tech 3	Q3Radiant /Gtk Radiant is a separate download.	Full source code available from id Software's ftp site.	Toolchain available for Linux to help with source compilation.	id Software provides documentation for level building,terrains, AI and the shader system.	Released in 1999, graphics look slightly dated, but still has an active community.
id Tech 4	D3Edit integrated with id Tech 4 based games.	Game source code available, comes with Doom 3 SDK.	Doom 3 SDK, includes Maya importer source, vehicle demo and sample Maya files.	An official id Software website provides an introductory guide.	Large community, good amount of tutorials and documentation available.
Jupiter EX	WorldEdit comes with F.E.A.R SDK.	Game source code available, comes with F.E.A.R SDK.	F.E.A.R SDK, includes documentation and additional level/game editing tools.	CHM file included with SDK gives an overview of the tools, some tutorials and some references.	F.E.A.R community not as large as for other games, but still provides some useful tutorials.
Source	Hammer comes with Source SDK.	Game source code available, comes with Source SDK.	Source SDK, includes additional tools. Requires Steam account.	Official documentation available on Valve Developer Community.	Large community providing comprehensive tutorials and references.
Unreal 2	UnrealEd, included with Unreal based games.	UnrealScript game source code available from UDN.	Karma physics SDK, but no separate SDK for the engine. Some tools available from UDN.	Subset of official documentation and tutorials available on the Unreal Developer Network (UDN).	Paid license required for full source and UDN access, but many tutorials available from community.

Table 1: Game engine comparison